Receiver-initiated Sending-rate Control based on Data Receive Rate for Ad Hoc Networks connected to Internet

Akihisa Kojima[†]and Susumu Ishihara[‡]

[†]Graduate School of Engineering, Shizuoka University
[‡]Graduate School of Science and Technology, Shizuoka University
3-5-1 Johoku, Naka-ku, Hamamtsu, 432-8561, Japan
(kojima|ishihara)@ishilab.net

ABSTRACT

We propose a scheme to improve TCP throughput on multihop wireless networks connected to the Internet. Though it changes the behavior of the receiver TCP, it does not change the sender TCP, in contrast to conventional techniques, which modify the sender TCP. Thus, our scheme can be useful even if a host on a wireless ad hoc network that is connected to the Internet communicates with web servers and mail servers that send most data to the host on the ad hoc network. In our scheme, the receiver TCP controls the timing of sending ACK segments using an algorithm similar to that used in TCP-VAR so that the sender TCP can send segments at an appropriate interval for the current condition of the network. Simulation results show that our scheme improves the throughput of TCP by an additional 40% compared with TCP-NewReno.

Keywords: Ad hoc network, TCP, rate control, simulation, TCP-VAR, multi-hop wireless network

1 INTRODUCTION

Obtaining high TCP throughput on wireless ad hoc networks is difficult due to interference and contentions as well as errors on wireless links and frequent changes of routes. Even if there is only one TCP flow on the network, packets of the same flow cause interference and contentions because multiple hosts send packets that are continuously sent from the source node, especially when the sender TCP enlarges its congestion window. This contention and interference is avoidable by using appropriate TCP-adaptive pacing (TCP-AP) [1] and TCP with variance control (TCP-VAR) [2], which are congestion avoidance techniques based on a similar idea.

However, TCP-AP and TCP-VAR need to change the TCP of the sender or of both the sender and the receiver. Consequently, applying these techniques is difficult when an ad hoc network is connected to the Internet and a user in the ad hoc network uses the web (Fig. 1). For example, most sender TCPs are on the Internet when a user accesses web servers on the Internet. TCPs of those servers are not likely to support peer hosts on wireless ad hoc networks. Thus, it is necessary to control the packet transmission interval through the receiver on the ad hoc network instead of the sender TCP outside the ad hoc network.

We propose a receiver-initiated sending-rate control scheme, RSC-VAR. In our scheme, the receiver TCP estimates a suitable transmission interval for the current condition of the ad hoc networks and controls the interval of ACK segments. To estimate a suitable transmission interval, we modified an algorithm used in TCP-VAR.

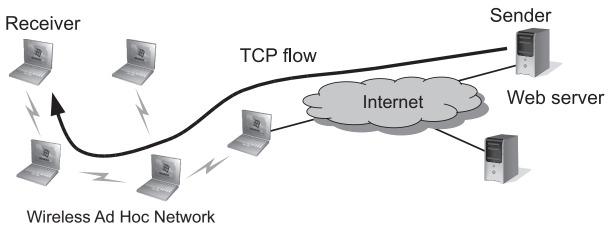


Figure 1: Ad hoc network connected to Internet

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 describes the details of the RSC-VAR, and Section 4 evaluates the scheme by simulation. Finally, we conclude the paper in Section 5.

2 RELATED WORK

Recently, several techniques have been proposed to improve TCP performance in multihop 802.11 wireless networks. Xu et al. proposed the neighborhood RED (NRED) scheme on the network layer to enhance TCP fairness [3]. The authors showed that NRED could substantially improve TCP fairness in multihop wireless networks. Fu et al. proposed a link-layer RED scheme to improve TCP performance by tuning the wireless link's drop probability to increase the spatial channel reuse [4]. ATCP, proposed in Ref. [5], distinguishes between packet loss by congestion and by bit error. It controls the packet transmission rate according to the reason for packet loss. ElRakabawy et al. proposed TCP-AP [1]. In that scheme, a TCP sender decides the data transmission interval in accordance with the four-hop propagation delay (FHD) and the delay variance on the wireless ad hoc network.

These techniques directly or implicitly interact with the lower layers. Other schemes that do not rely on cross layer information to improve TCP performance in wireless networks have also been proposed. In Ref. [6], the use of a small congestion window (cwnd) probing rate is proposed. In the scheme, the sender slows down the cwnd probing rate to make TCP less aggressive in wireless networks. Because the bandwidth delay product in wireless networks is small, the scheme

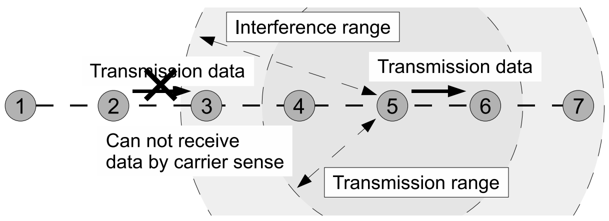


Figure 2: Self Contention: Assuming a TCP flow from node 1 to node 7, and that node 5 transmits data packet to node 6, collisions occur between nodes 2 and node 3 because node 3 listens to the transmission by node 5.

improves TCP efficiency by moderated TCP aggressiveness leading to reduced routing reaction to packet losses at the MAC layer. Chen et al. proposed TCP-VAR, in which the sender TCP estimates the sending rate depending on the fluctuation of the achieved throughput [2].

TCP-AP and TCP-VAR improve TCP performance by estimating an appropriate transmission interval. However, these schemes need to modify the sender TCP. Thus, they are not useful when a user uses the web in ad hoc networks connected to the Internet because the sender TCPs on general web servers on the Internet are not likely to support functions for peers on ad hoc networks. Gateway TCP-AP [7] is an enhancement of TCP-AP for ad hoc networks connected to the Internet. However, it is also a sender-side solution and uses a special gateway.

In this paper, we propose a receiver-side scheme for improving TCP performance on ad hoc networks connected to the Internet. Altman proposed a receiver-side scheme named Dynamic Delayed ACK[8]. It avoids contention between data packets and ACK packets by dynamically changing delayed ACK parameter d. In other words, this scheme reduces the number of ACK packets. In our scheme, the receiver TCP controls the timing of sending ACKs to control the interval of transmission of data packets from the sender TCP.

3 TCP TRANSMISSION INTERVAL ON AD HOC NETWORKS

In this section we explain the motivation for expanding the transmission interval to improve TCP performance on ad hoc networks, and we describe TCP-VAR as a basic idea of estimating the sending rate.

3.1 The Motivation for Expanding Transmission Interval

Besides the measure of contention on the network path, the derivation of an appropriate transmission rate should also account for the spatial reuse constraint of IEEE 802.11 multi-hop wireless networks. In a chain topology such as in Fig. 2, a TCP flow starts at node 1 and travels on the chain to node 7. Assume that node 2 wishes to transmit data to node 3 and node 5 wishes to transmit data to node 6. In this topology, node 5 is hidden from node 2. Thus, node 5 may transmit a frame to node 6 while node 2 is transmitting a frame to node 3. This causes a collision at the receiving node 3, because node 3 is within the interference range of node 5. We call such a contention a self-contention. Such self-contention occurs when the transmission interval is short, e.g., for example when the sender increases the size of congestion window. To solve this problem, Chen proposed TCP-VAR which dynamically detects the optimal interval from the throughput fluctuation.

3.2 TCP with Variance Control (TCP-VAR)

In TCP-VAR, the sender TCP controls the packet transmission interval according to the throughput fluctuation level. A small throughput fluctuation means that the network is not congested and TCP can increase the sending rate: if otherwise, the network is congested and TCP should throttle the sending rate. TCP-VAR controls the sending rate by changing the special congestion window, which the authors call virtual cwnd (vcwnd), and determines the sending rate according to the current vcwnd, the round trip time (RTT), and the throughput fluctuation level.

To quantify the fluctuation of achieved throughput, the sender calculates the fluctuation of the recently measured achieved throughput samples. The achieved throughput is defined as ACKed data in a recent period:

$$AT = \frac{S_{\text{acked}}}{T_{\text{srtt}}}, \tag{1}$$

where AT, T_{srtt} and S_{acked} represent the achieved throughput, the smoothed RTT (sRTT), and the size of acked data in the last sRTT respectively.

The average and fluctuation of AT (AVE_{AT}, VAR_{AT}) are defined based on K recent samples of AT.

$$AVE_{AT} = \frac{\sum_{i=1}^{K} AT_i}{K} \tag{2}$$

$$VAR_{AT} = \frac{\sum_{i=1}^{K} AT_i - AVE_{AT}}{(K-1) \cdot AVE_{AT}} \tag{3}$$

TCP-VAR takes the following linear formula to control the outgoing packet rate:

$$R = \frac{W_{\text{vc}}}{T_{\text{srtt}} \cdot (1 + \exp(VAR_{AT}))}, \tag{4}$$

where R and W_{vc} represent the outgoing TCP sending rate at the source and virtual congestion window respectively. The reasoning behind this formula is to penalize the rate in high fluctuation scenarios. The sending rate should be penalized when the fluctuation is high.

When the fluctuation is continuously high, vcwnd is decreased and the sending rate drops as well. When the fluctuation is continuously low, vcwnd is increased. Finally, for a packet loss, TCP-VAR halves vcwnd.

In our scheme, RSC-VAR, the receiver TCP estimates the suitable interval of ACKs for controlling the interval of data packets from the sender TCP using a similar technique to that used in TCP-VAR.

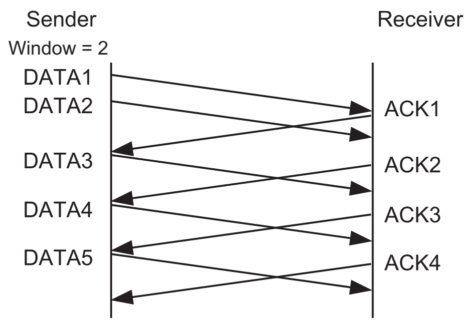


Figure 3: Controlling data packet transmission interval by ACK packets.

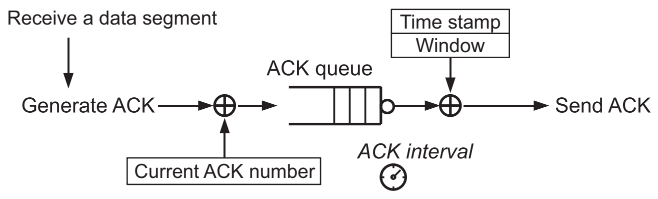


Figure 4: Adaptive ACK Pacing: If a receiver receives a data segment, it generates an ACK for the data segment with the current ACK # and puts it in the ACK queue. The receiver sends the ACK at the top of the ACK queue when the timer expires.

4 RECEIVER-INITIATED SENDING-RATE CONTROL BASED ON TCP-VAR (RSC-VAR)

We next present a detailed description of RSC-VAR. In the scheme, the receiver TCP controls the ACK transmission interval and the advertisement window in order to control the sending rate of the sender TCP. In the following description, we assume that the size of all data segments is the same as the maximum segment size and that the sender has enough large data to send.

4.1 ACK Transmission Control

Generally, receiving an ACK segment of TCP triggers transmission of a data segment from the TCP sender because the ACK indicates the slide of the sending window. If the receiver delays sending an ACK, the sender also delays sending a new data segment.

To control the timing of the transmission of ACKs, in RSC-VAR, the receiver TCP buffers ACKs generated by receiving data segments. We name the buffer an ACK queue. When the receiver TCP receives a data segment, it generates an ACK segment by using the same algorithm as conventional TCPs such as TCP-NewReno. The receiver TCP puts the ACK segment into the ACK queue instead of sending it immediately. ACKs put into the queue are sent to the sender TCP when a timer for controlling the transmission timing of ACKs expires. In this way, an ACK segment acknowledges the successful transmission of data segments at the controlled interval. Therefore, the receiver can control the sending rate of the sender if the sender's congestion window is fixed.

However, the size of the sender's congestion window is changed according to the congestion avoidance algorithm used by the sender TCP. The receiver can not change the value directly. Instead, the receiver controls the upper limit of the sending window of the sender by controlling the advertisement window on ACK segments. The receiver estimates the bandwidth delay product based on the ACK interval and the smoothed round trip time and reports it to the sender by using the advertisement window.

The receiver calculates the advertisement window as follow:

$$W_{\mathrm{adv}} = \frac{T_{\mathrm{srtt}}}{T_{\mathrm{ack}}} \cdot S_{\mathrm{mss}} \tag{5}$$

where W_{adv} is the value of the advertisement window, T_{ack} is the ACK interval and S_{mss} is the maximum segment size.

4.2 Calculation of Sending Rate by a Receiver

The receiver determines the transmission interval based on the fluctuation of the achieved throughput. RSC-VAR makes the ACK interval shorter when the fluctuation level is low and makes it longer when the fluctuation level is high.

The receiver calculates the outgoing TCP sending rate (R) at the sender TCP according to the ACK interval:

$$R = \frac{1}{T_{\mathrm{ack}}} \tag{6}$$

T_{ack} is calculated as follow:

$$T_{\mathrm{ack}} = T_{\mathrm{base}} \cdot \exp(VAR_{AT}) \tag{7}$$

Let T_{base} be the base value of the ACK interval that RSC-VAR changes based on the fluctuation level of the achieved throughput VAR_{AT} defined by Eq. 3. This formula penalizes the rate in high fluctuation scenarios in a similar way to TCP-VAR (Eq. 4). However, the definition of AT is different from TCP-VAR. RSC-VAR defines AT as follows:

$$AT = S_{\mathrm{data}}/T_{\mathrm{srtt}}, \tag{8}$$

where S_{data} is the size of data received by the receiver TCP within the last smoothed round trip time (sRTT). The sample of AT is calculated when the receiver TCP receives a new data segment.

We dynamically change K according to the fluctuation level VAR_{AT}. K is the number of samples of achieved throughput to calculate the fluctuation (Eq. (3)). If the fluctuation level is higher than the threshold (θ_{co}), K is set to a small value ($K_{\mathrm{congestion}}$). If the network is congested, the receiver calculates the fluctuation of the achieved throughput for a small number of samples. This is to quickly react to the changes of the network condition. Otherwise, large K ($= K_{\mathrm{normal}}$) is used to measure the long-term fluctuation of the achieved throughput.

In RSC-VAR, the receiver changes T_{base} based on the fluctuation level. When the fluctuation of AT is higher than the threshold, T_{base} is decreased and the sending rate becomes high. On the other hand, T_{base} is increased when $\overline{VAR_{AT}}$ is lower than the threshold. RSC-VAR defines two states for T_{base} control, slow start (SS) and congestion avoidance (CA). T_{base} is calculated according to the following expressions:

$$T_{\text{base}} = \begin{cases} \alpha \cdot T'_{\text{base}} & (s = \text{SS} \wedge \overline{VAR_{AT}} < \theta_{\text{ss,low}}) \\ \beta \cdot T'_{\text{base}} & (s = \text{SS} \wedge \overline{VAR_{AT}} > \theta_{\text{ss,high}}) \\ T'_{\text{base}} - \gamma & (s = \text{CA} \wedge \overline{VAR_{AT}} < \theta_{\text{ca,low}}) \\ T'_{\text{base}} + \delta & (s = \text{CA} \wedge \overline{VAR_{AT}} > \theta_{\text{ca,high}}) \end{cases} \quad (9)$$

where T'_{base} and s mean previous T_{base} and the current state of the receiver respectively. $\theta_{\text{ss,low}}$, $\theta_{\text{ca,low}}$ are the thresholds to decrease T_{base}. $\theta_{\text{ss,high}}$, $\theta_{\text{ca,high}}$ are the thresholds to increase T_{base}. α, β, γ, and δ are constant values ($0 < \alpha < 1$, $\beta > 1$, $0 < \gamma$, $0 < \delta$).

4.2.1 How to Initialize T_{base}

The initial value of T_{base} should be long enough for self contention to be avoided. Thus, the initial value of T_{base} is set to half of the round trip time (RTT). However, RTT can not be estimated before the communication starts. Therefore, in RSC-VAR, the receiver behaves like a normal TCP receiver until the first RTT is measured.

4.2.2 Timing of Calculating T_{base}

The receiver calculates the fluctuation of the achieved throughput and ACK interval whenever it receives a data segment. However, T_{base} is updated every two sRTTs. This is because if the receiver calculates T_{base} whenever it receives a data segment, T_{base} changes too frequently.

For example, let us consider a case where the RTT is very long, but the wireless network is not congested, and the T_{base} is updated whenever the receiver TCP receives a data segment. In this case, when the receiver receives a data segment, it decides to decrease T_{base} because the calculated fluctuation of the achieved throughput is low. The ACK interval (T_{ack})is immediately made short according to Eq. 7. However, the time until the sender changes the data transmission interval is long due to the long RTT.

Within the time lag between the calculation of the small T_{base} and the change of the data sending at the sender, the receiver receives other data segments and calculates the fluctuation of the achieved throughput. The result of the calculation of the fluctuation will be low because the received data segments are still sent at a low rate. Accordingly, if T_{base} changes whenever the receiver receives a data segment, T_{base} becomes too short. The fluctuation level then becomes high due to the too short T_{base}, so T_{base} increases. When RTT is long, the performance will worsen due to the frequent update of T_{base}.

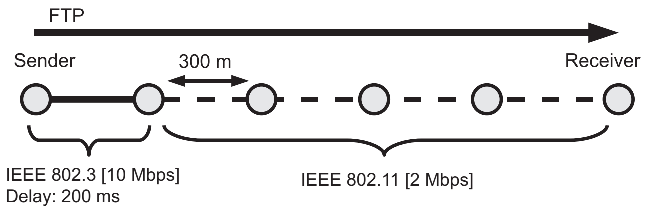


Figure 5: Chain topology: 1-hop wired link and 4-hop wireless links

4.2.3 Slow Start and Congestion Avoidance

RSC-VAR defines two states for T_{base} control: slow start and congestion avoidance. Just after the set up of the connection, the state is slow start until the sending rate grows, i.e., T_{base} becomes small enough. Then it goes to the congestion avoidance state. In the slow start state, T_{base} is decreased exponentially according to Eq. 9. In this state, the fluctuation of the achieved throughput tends to be big because T_{base} varies dramatically. Accordingly, big thresholds ($\theta_{\text{ss,low}}$, $\theta_{\text{ss,high}}$) are set. In the congestion avoidance state, T_{base} is decreased linearly and small thresholds ($\theta_{\text{ss,low}}$, $\theta_{\text{ss,high}}$) are set.

The state changes to congestion avoidance from slow start when a receiver detects evidence of congestion or the approach of the ACK interval to the optimal value. The receiver detects congestion by the fluctuation level or the interval between receiving data segments. The fluctuation level becomes high when the network becomes congested. The interval of incoming data becomes large when packets are dropped extensively, and the receiver detects it by the time out. If the ACK interval becomes an optimal value, T_{base} does not change. Accordingly, if T_{base} is the same value for a certain period, the ACK interval is optimal.

4.2.4 Handling Packet Drops

Traditional TCPs treat packet drops as the evidence of congestion and consequently throttle their congestion window. Similar to conventional TCPs, the RSC-VAR increases the ACK interval ($T_{\text{base}} = \beta \cdot T'_{\text{base}}$) when it detects packet drops. However, it ignores packet drops without congestion. A receiver of RSC-VAR increases the ACK interval only when it detects a packet drop by a time out , which indicates that data segments have not arrived for a long time.

5 PERFORMANCE EVALUATION

5.1 Simulation Environment

We conducted a simulation of the proposed scheme using QualNet 4.0 with the following configurations. The IEEE 802.11b standard was adopted where the transmission range was about 340 m and the carrier sensing range was about 540 m. The transmission of each data packet on the MAC

Table 1: Parameters of RSC-VAR

$\theta_{\mathrm{ss,high}}$	0.3	n	50
$\theta_{\mathrm{ss,low}}$	0.2	α	0.9
$\theta_{\mathrm{ca,high}}$	0.15	β	1.2
$\theta_{\mathrm{ca,low}}$	0.05	γ	1 ms
θ_{co}	0.3	δ	3 ms
K_{normal}	50	TimeOut	$4 \cdot sRTT$
$K_{\mathrm{congestion}}$	5		

layer was preceded with a Request-To-Send/Clear-To-Send (RTS/CTS) handshake. The channel bandwidth was 2 Mbps.

The wired bandwidth was 10 Mbps, and delay was 200 ms. In all scenarios, the sender TCP was NewReno, and the packet size was 1460 bytes. No routing protocols were used and the routing tables in each node were static.

The duration of each simulation was 300 s. Node 1 in Fig. 5 sends data to node 6 by FTP during the scenario. We present a comparative performance study of RSC-VAR versus NewReno at the receiver TCP on the ad hoc network.

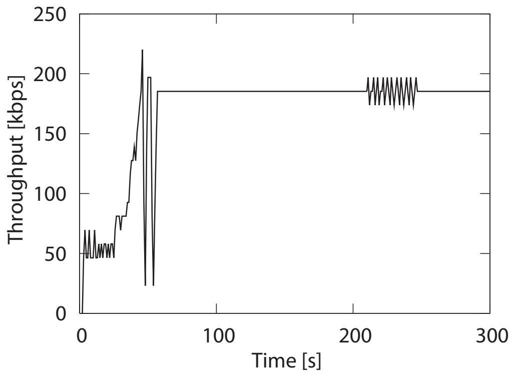
We set the parameters of RSC-VAR as shown in Table 1.

5.2 Simulation Results

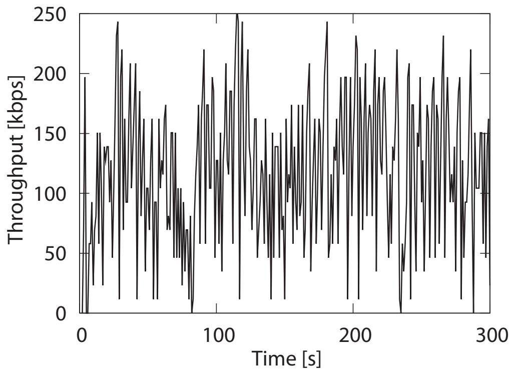
We ran the simulator 30 times for each configuration. The throughput of TCP flows whose TCP receivers are RSC-VAR and NewReno are sown in Fig. 6. The shown throughput in the figure is a result of a good case simulated 30 times. As expected, RSC-VAR showed smoother instantaneous throughput than NewReno's. The average throughput of each simulation is shown in Fig. 7. The result is sorted so that the smaller throughput values are to the left. Figure 7 shows that RSC-VAR achieved higher throughput than NewReno in good cases. However, in bad cases, RSC-VAR degraded throughput. As shown in Fig. 6(a), in a good case, RSC-VAR maintains high and stable throughput 60 s after the beginning of the session. The average throughput of RSC-VAR in the simulation was about 167 kbps. The throughput of NewReno instantaneously achieved high throughput. However, overall, it fluctuated extensively. The average throughput was about 119 kbps, about 70% of that of RSC-VAR.

An example of the change of ACK interval is shown in Fig. 8. The broken line in Fig. 8 is the threshold of determining the congestion. The ACK interval decreased when the receiver was in the slow start state until 50 seconds after the beginning of the session. Then, the ACK interval increased due to time outs which indicate packet drops. After 60 seconds, the ACK interval was stable at about 55 ms which was slightly longer than the threshold.

The average packet loss ratio in the simulations is shown in Table 2. The number of packet losses of RSC-VAR was less than NewReno. The non-aggressive nature of RSC-VAR did not place a heavy burden on the MAC layer. However, RSC-VAR sent 7554 packets, which was more than NewReno did (6685), in spite of extending the data-sending interval.



(a) RSC-VAR



(b) NewReno

Figure 6: Throughput on a 4-hop wireless chain with 1-hop wired link

Table 2: Packet Loss Rate

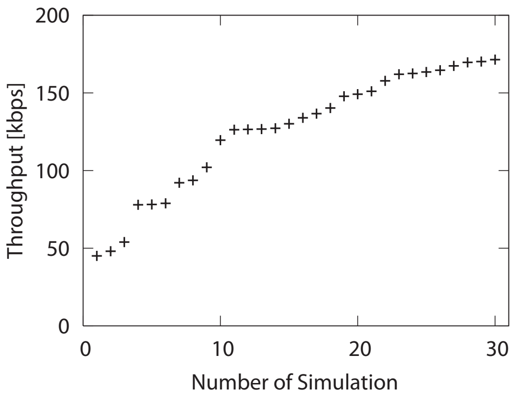
	Packet loss rate [%]	Loss	Total
RSC-VAR	0.053	4	/ 7554
NewReno	3.680	246	/ 6685

5.3 Discussion

The result of a good case among 30 simulations for one configuration is shown in Fig. 6. Some of the simulations resulted in bad throughput as represented by the solid curve in Fig. 9. In this case, a congestion occurred at about 40 seconds and the throughput was suppressed in the subsequent period. The reason was a failure to detect congestion by the fluctuation level. After 40 seconds, the sending rate was low enough to avoid congestion. However the fluctuation level was too high to decrease T_{base}. One of the reasons for this was the calculation of the fluctuation of the achieved throughput and the value of the threshold for controlling T_{base}. The fluctuation tends to be high when the average of AT is low. Hence, the throughput was suppressed after 40 seconds (Fig. 9).

6 CONCLUSION

We proposed a receiver-initiated congestion control scheme for TCP on ad hoc networks connected to the Internet. RSC-


(a) RSC-VAR

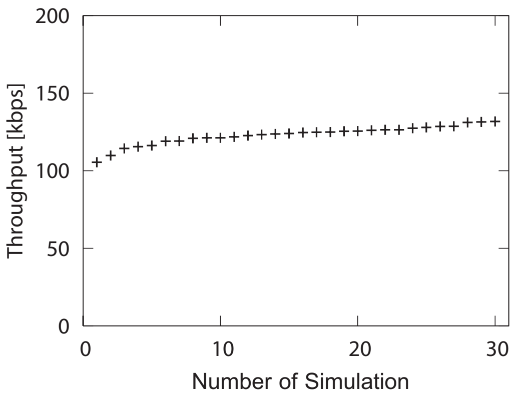

(b) NewReno

Figure 7: Average throughput of each simulation

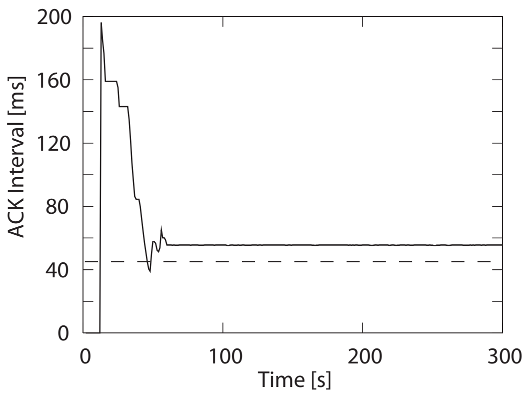


Figure 8: ACK interval of RSC-VAR

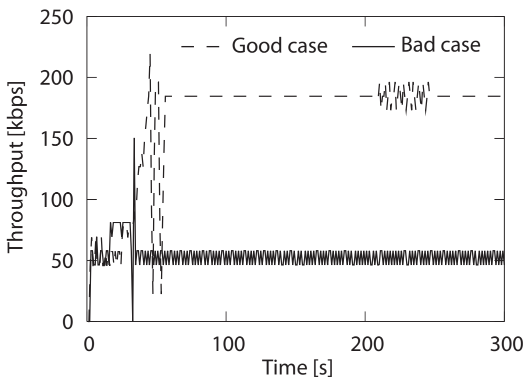


Figure 9: Throughput of RSC-VAR in a good case and bad case

VAR controls the ACK transmission interval at a receiver to control the data transmission interval at the sender according to the fluctuation of the achieved data throughput. Because RSC-VAR does not rely on the mechanism at the sender TCP, it can be used to improve the performance of TCP between an arbitrary TCP host on the Internet and a host in an ad hoc network connected to the Internet. The simulation results showed that it improves average TCP throughput upto an additional 40% compared to using the TCP-NewReno receiver. However, in bad cases, RSC-VAR significantly degrades the throughput. Further analysis of the behavior of RSC-VAR and improvement of the mechanism will be studied in the future.

REFERENCES

[1] S. M. ElRakabawy, A. Klemm, and C. Lindemann, “TCP with adaptive pacing for multihop wireless networks,” in *Proceedings of ACM MobiHoc’05*, 2005, pp. 288–299.

[2] J. Chen, M. Gerla, Y. Z. Lee, and M. Sanadidi, “TCP with variance control for multihop ieee 802.11 wireless networks,” in *Proceedings of IEEE MILCOM’06*, Oct. 2006, pp. 1–7.

[3] K. Xu, M. Gerla, L. Qi, and Y. Shu, “Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED,” in *Proceedings of ACM MobiCom’03*, 2003, pp. 16–28.

[4] A. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, “The impact of multihop wireless channel on TCP throughput and loss,” in *Proceedings of IEEE INFOCOM’03*, vol. 3, Apr. 2003, pp. 1744–1753.

[5] J. Liu and S. Singh, “ATCP: TCP for mobile ad hoc networks,” *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1315, July 2001.

[6] K. Nahm, A. Helmy, and C.-C. J. Kuo, “TCP over multihop 802.11 networks: issues and performance enhancement,” in *Proceedings of ACM MobiHoc’05*, 2005, pp. 277–287.

[7] S. M. ElRakabawy, A. Klemm, and C. Lindemann, “Gateway adaptive pacing for TCP across multihop wireless networks and the internet,” in *Proceedings of ACM MSWiM’06*, 2006, pp. 173–182.

[8] E. Altman and T. Jimenez, “Novel delayed ACK techniques for improving TCP performance in multihop wireless networks,” in *Proceedings of PWC’03*, Sep. 2003, pp. 237–253.